08

El estándar MVC

Transcripción

[00:00] Hola. Ahora que estamos prácticamente con nuestra aplicación completa, vamos a realizar un poco el proyecto y entender que estuvimos haciendo hasta ahora. Después que configuramos la base de datos, realizando la instalación de MySQL y creando la base de datos de nuestro proyecto de control de stock, aprendimos cómo configurar nuestra aplicación Java.

[00:20] Vamos a agrandar un poquito la letra. Aprendimos a conectar nuestra aplicación Java a la base de datos de MySQL por medio de algunas librerías, que son la de JDBC y la de las principales, el driver de MySQL. Aquí tenemos las flechitas y de esta forma después de la instalación de MySQL aprendimos a conectar nuestra aplicación Java con la base de datos por medio de estas librerías.

[01:01] Después de esto nosotros aquí en Eclipse importamos un super proyecto, que a primera vista parecía complejo pero que de a poco fuimos aprendiendo cómo caminar por su código e implementar las funcionalidades que faltaban para darle vida y conectarlo con la base de datos.

[01:16] Este es un proyecto desarrollado en Java Swing, es la forma de desarrollar vistas de aplicación, así como el HTML para aplicaciones web. Pero la diferencia aquí es que ella no corre en un servidor de aplicaciones, sino en nuestra propia máquina, un ejecutable.

[01:32] Es una aplicación embebida. Hasta ahí todo bien. Ahora vamos a revisar un poco los componentes de esta aplicación. Nosotros cuando inicializamos

aplicación en el Main, nosotros llamamos a esta clase de ControlDeStockFrame que tiene un constructor que contiene todo el código que crea nuestra pantalla, que enlista y registra los productos. La pantalla es esta de acá que ya nos está acompañando en todo el curso.

[02.00] Esa pantalla acá tiene el formulario y el listado de productos que es construida por el ControlDeStockFrame, es responsable por presentar al usuario las informaciones buscadas en la base de datos de una forma ordenada. Esto aquí compone nuestra capa de view, que es la vista de la aplicación.

[02:18] Cada botón que tenemos aquí en la pantalla tiene una acción configurada y estas acciones ejecutan un conjunto de métodos. Por ejemplo, si entramos aquí en configurar acciones de formularios y vemos la acción del botónGuardar, este botón de guardar llama al método guardar, está aquí adentro, limpiar la tabla y cargar la tabla.

[02:40] Este método de guardar, ¿qué hace? Toma las informaciones del formulario y crea un objeto del tipo producto. Este objeto del tipo producto es el que representa nuestra tabla de producto en la base de datos, pero aquí en el proyecto de Java.

[02:58] Luego de eso, cuando crea el producto, lo envía para el productoController en el método guardar. La clase productoController también tiene las demás operaciones que nuestra lista ejecuta, como la de listar, eliminar y modificar. Aquí habíamos empezado agregando toda aquella lógica para abrir la conexión, ver la query, ejecutar la operación, devolver el resultado pero ahora tenemos solamente llamadas a métodos de la clase de productosDAO.

[03:31] El productoController pertenece a la capa de controller, que es la capa que hace la conexión de la vista con la capa de datos y contiene las lógicas de

negocio para manipular los datos antes de guardar en la base de datos o para devolver a la pantalla.

[03:47] Por último tenemos aquí la clase de productoDAO que es la que contiene toda la lógica relacionada a operaciones de la base de datos con la conexión, con la creación de queries, con la conversión de un objeto para hacer la query para insert, para update o delete o también para tomar el resultado y convertir en result set en un objeto del tipo producto para devolver a la pantalla.

[04:15] Como había comentado tenemos todas las operaciones de alta, baja, modificación y de listado. La clase productoDAO tiene la finalidad de realizar las operaciones directas en la tabla de producto. Entonces ella tiene una conexión directa con el modelo de producto.

[04:34] Si nosotros llegamos a tener nuevas tablas en la aplicación, nosotros vamos a crear nuevas clases DAO y nuevas clases de modelo también para representar a estas tablas en la aplicación y para realizar las operaciones sobre ellas.

[04:48] El conjunto de clases de modelo, producto y de la clase productoDAO, forman nuestra capa de modelo, la model, que representa las entidades del negocio y realiza las operaciones sobre sus informaciones. Para este conjunto de capas que revisamos ahora, le damos en nombre de modelo MVC, de Model View Controller.

[05:10] Este modelo es un estándar de arquitectura de aplicación que ayuda a dividir las responsabilidades de una aplicación. Y estas responsabilidades están divididas en las tres capas que recién conocimos. Este modelo tiene como ventajas, más allá de la división de las responsabilidades, la facilidad de mantenimiento, claridad y reutilización del código.

[05:31] ¿Por qué tenemos que utilizar aquí la capa de controller si no tenemos ninguna lógica acá? Nosotros solamente enviamos todo lo que recibimos para

la clase de productoDAO. Podríamos aquí, en el ControlDeStockFrame llamar directamente el productoDAO haciendo las operaciones directo de la View.

[05:51] Bueno, podríamos hacer eso, pero eso no es una buena práctica, porque terminamos creando una relación entre dos estructuras y tiene sus responsabilidades bien definidas. La vista debe mostrar la información devuelta por la base de datos y el DAO debe representar el modelo y realizar las operaciones que conecten la aplicación a la base de datos.

[06:14] Si para realizar la requisición desde la view hay una lógica que involucra más de una clase de modelo por detrás, ¿cuál de las dos capas debería tener la responsabilidad? ¿Deberíamos poner todo acá en la view o deberíamos agregar todo acá en el DAO? Ninguna de ellas. Por eso es que tenemos aquí la capa de productoController.

[06:37] Porque ella tiene su importancia en este caso porque ella, más allá de realizar esta conexión entre la vista y el modelo, ella también realiza las operaciones relacionadas a las reglas de negocio para completar una requisición. Entonces si nosotros tenemos aquí la entidad de producto y queremos relacionarla a una otra entidad, nosotros podremos hacer la operación directamente aquí en productoController y no impactaría la finalidad de ninguna de las otras dos capas.

[07:05] Así que, por más sencilla que sea la clase de productoController, su presencia tiene gran importancia justamente porque si el proyecto evoluciona, es en ella que empezaremos a agregar más lógicas de negocio. Espero que les haya gustado entender un poco más del concepto de lo que venimos desarrollando.

[07:24] El modelo MVC es sencillo y aún sigue siendo muy adoptado por empresas para desarrollar aplicaciones del mundo real. Hay otros modelos de arquitectura y variaciones de cada uno que pueden ser utilizados. Cada uno

con sus ventajas y objetivos. Nos vemos en la próxima clase para realizar algunas mejoras finales en el proyecto y desarrollar una última funcionalidad.